Claims

1. A method of efficiently using computing resources including multiple microprocessors and at least one memory, said method comprising the steps of;
   - a first process requesting a task,
   - a second process determining the resources available for servicing the task, such resources comprising a first microprocessor running said first and second processes and a second microprocessor,
   - said second process determining the capabilities of said second microprocessor,
   - said second process optimizing said task for ultimate computation,
   - said second process determining if said optimized task is within said determined capabilities of said second microprocessor,
   - compiling said task for said second microprocessor if a determination has been made that said second microprocessor has capabilities to run said compiled task.
2. The method of claim 1 further comprising the step of compiling said task for said first microprocessor if said second microprocessor has been determined to have insufficient capabilities.
3. The method of claim 2 wherein said compiling said task for said first microprocessor include emulation.
4. The method of claim 1 further comprising the step of running the task on said first microprocessor by emulating said second microprocessor on said first microprocessor.
5. The method of claim 1 wherein said first process is an application program.
6. The method of claim 1 wherein said second process is a graphics services resource suite.
7. The method of claim 1 wherein an operating system comprises said second process.
8. The method of claim 1 wherein said first microprocessor comprises a CPU.
9. The method of claim 1 wherein said second microprocessor is a GPU.
10. The method of claim 8 wherein said second microprocessor is a GPU.
11. The method of claim 1 wherein said second process determines the status of said second microprocessor.

12. The method of claim 1 wherein said second process determines the status of at least one memory resource.

13. The method of claim 1 further comprising the step of running said compiled task on said second microprocessor.

14. A method of creating an image in a computer system, comprising the steps of :

- A second process receiving a request from a first process to create said image, said image having an object associated therewith;
- Said second process determining whether said computer system has a suitable GPU accessible to said second process;
- If a suitable GPU is accessible, said second process optimizing said object and compiling for rendering on said GPU;
- If a suitable GPU is not accessible, said second process optimizing said object and emulating for rendering on a CPU.

15. The method of claim 14 wherein said object comprises a graph representing said image.

16. The method of claim 14 wherein said image is created by applying one or more filters to a pre-existing image.

17. The method of claim 14 wherein said object comprises program instructions including reference to all items necessary for creating said image.

18. The method of claim 14 wherein said suitable GPU is a programmable GPU.

19. The method of claim 14 wherein said suitable GPU is accessible if said second process may access the functionality of said GPU through another program.

20. The method of claim 14 wherein said suitable GPU is not accessible if there is no GPU in said computer system.

21. The method of claim 14 wherein said suitable GPU is not accessible if there is no programmable GPU in said computer system.

22. The method of claim 14 wherein said suitable GPU is not accessible if there is no programmable GPU in said computer system that is compatible with said second process..

23. The method of claim 14 wherein said suitable GPU is not accessible if there is a programmable GPU in said computer system but it is busy.

24. The method of claim 14 wherein said suitable GPU is not accessible if there is a programmable GPU in said computer system but said second process does not have authority to access said GPU.

25. A method of creating an image in a computer system, comprising the steps of:
   - Receiving a request to render a polygon;
   - Converting a GPU program for rendering said polygon into a CPU program;
   - Dividing said polygon into sections;
   - Selecting a first section for rendering;
   - Selecting a first scan line in said first section for rendering;
   - Rendering at least a portion of said scan line in increments of L number of pixels,
   - Interpolating texture values for at least some of said pixels.

26. The method of claim 25 wherein said GPU program is expressed as a graph.

27. The method of claim 26 wherein said graph is optimized before converting to said CPU program.

28. The method of claim 27 wherein optimizing comprises the step of performing a global intersection of DOD and ROI.

29. The method of claim 27 wherein optimizing comprises the step of combining fragment programs.

30. The method of claim 27 wherein optimizing comprises the step of attempting to combine fragment programs.

31. The method of claim 25 comprising the further step of checking a cache to determine if said GPU program may be retrieved from memory.

32. The method of claim 25 wherein said CPU program comprises a main loop for which a single execution of said main loop will process N number of pixels.

33. The method of claim 32 wherein said main loop processes L number of pixels by running the main loop program L/N times.

34. The method of claim 32 wherein a second scan line has a number of pixels S that is E pixels greater than the greatest multiple of L that is less than S, the method further comprising the steps of:
   - Running the code in said main loop on successive groups of L pixels until there are E pixels left to process;

- Adjusting a variable in said main loop to cause said main loop run R number of times processing R times N pixels, where R times N is the highest multiple of N that is less than or equal to E.

35. The method of claim 34 further comprising the step of running said adjusted main loop.

36. The method of claim 32 wherein

- Said CPU program comprises a routine for processing one pixel, said processing analogous to that done in the main loop, and
- a second scan line has a number of pixels S that is E pixels greater than the greatest multiple of L that is less than S,
- the method further comprising the steps of:
  - Running the code in said main loop on successive groups of L pixels until there are E pixels left to process;
  - Running said routine for processing one pixel E times.

37. The method of claim 35 wherein

- Said CPU program comprises a routine for processing one pixel said processing analogous to that done in the main loop, and
- the method further comprising the step of;
  - Running said routine for processing one pixel a number of times to process any pixels in the scan line not yet processed.

38. The method of claim 25 where said sections are slabs.

39. The method of claim 25 where said sections are triangles.

40. The method of claim 25 wherein said first section is rendered using a first CPU and a second section is rendered by a second CPU.

41. The method of claim 25 wherein said first scan line is rendered using a first CPU and said second scan line is rendered using a second CPU.

42. The method of claim 33 wherein said CPU program performs independent texture lookups for each group of L pixels before processing any of said L pixels..

43. The method of claim 33 wherein said CPU program performs independent texture lookups for each said section before processing any pixels in said each section.

44. The method of claim 33 wherein function calls are placed in said CPU program for each dependent texture reference, said function calls performing the texture lookup during rendering.

45. The method of claim 43 wherein function calls are placed in said CPU program for each dependent texture reference, said function calls performing the texture lookup during rendering

46. A method of creating an image comprising the steps of:
   - Receiving a request to render a polygon;
   - Converting a GPU program to a CPU program by converting GPU instruction lines to one or more CPU instruction lines;
   - Unrolling a loop in said CPU program such that the unrolled loop will process N pixels;
   - Setting said unrolled loop to execute a variable amount of times, said variable V;
   - Rendering pixels in groups of L pixels, by executing said unrolled loop L/N times;

47. The method of claim 46 wherein said CPU program performs independent texture lookups for each group of L pixels before processing any of said L pixels.

48. The method of claim 46 further comprising the steps of
   - Dividing said polygon into sections;
   - Selecting a first section for rendering;
   - Selecting a first scan line in said first section for rendering;

49. The method of claim 48 wherein said CPU program performs independent texture lookups for each group of L pixels before processing any of said L pixels.

50. The method of claim 48 wherein said CPU program performs independent texture lookups for each said section before processing any pixels in said each section.

51. The method of claim 46 wherein function calls are placed in said CPU program for each dependent texture reference, said function calls performing the texture lookup during rendering.

52. The method of claim 48 wherein function calls are placed in said CPU program for each dependent texture reference, said function calls performing the texture lookup during rendering

53. The method of claim 49 wherein function calls are placed in said CPU program for each dependent texture reference, said function calls performing the texture lookup during rendering.

54. The method of claim 50 wherein function calls are placed in said CPU program for each dependent texture reference, said function calls performing the texture lookup during rendering

55. The method of claim 46 wherein all of L rendered pixels are stored in a buffer before delivering those pixels to a requested context.

56. The method of claim 48 wherein all of L rendered pixels are stored in a buffer before delivering those pixels to a requested context

57. The method of claim 48 wherein all of rendered pixels owing to one of said sections are stored in a buffer before delivering those pixels to a requested context.

58. A computer-readable medium having computer executable instructions for performing the method recited in any one of claims 1, 14, 25, 34, 36, or 46.

59. A method of creating an image comprising the steps of:

- receiving a request to render an image;
- creating an object representing said image, said object comprising one or more programs, said one or more programs representing GPU fragment programs for a programmable GPU;
- running a compiled version of said object using a virtual machine under control of a CPU.

60. The method of claim 59 wherein compilation comprises the steps of:

- Unrolling a loop in a GPU program such that the unrolled loop will process N pixels;
- Setting said unrolled loop to execute a variable amount of times, said variable V;
- performing independent texture lookups for each group of L pixels before processing any of said L pixels.
- Rendering pixels in groups of L pixels, by executing said unrolled loop L/N times;

61. The method of claim 59 comprising the additional steps of:

- Rendering pixels in groups of L;
- Creating one or more software registers having width L.

62. The method of claim 59 wherein said virtual machine accepts byte-code as input

63. The method of claim 59. compiled version of said object is ready for execution by said programmable GPU. 59.

64. A method of creating a result image comprising the steps of:
   - receiving a request to render said result image;
   - creating an object representing said result image, said object comprising one or more programs for applying at least one filter to at least one image, said one or more programs representing GPU fragment programs for a programmable GPU;
   - running a representation of said object on a CPU through emulation.

65. The method of claim 64 wherein said at least one filter is a high-level filter.

66. The method of claim 64 wherein said object is a high-level graph.

67. The method of claim 64 wherein said object is a low level graph.

68. The method of claim 64 wherein said one or more programs are fragment programs.

69. The method of claim 64 wherein said one or more programs are graph nodes.

70. The method of claim 64 wherein said representation of said object is a compiled version of said object.

71. The method of claim 64 wherein a virtual machine is used to run said representation of said object.

72. The method of claim 70 wherein a virtual machine is used to run said representation of said object.

73. The method of claim 64 wherein a byte-code virtual machine is used to run said representation of said object.

74. The method of claim 70 wherein a byte-code virtual machine is used to run said representation of said object.

75. The method of claim 64 wherein said representation of said object is a CPU program converted from said object line-by-line.

76. A method of applying two effects to an image, the method comprising the steps of
   - using a first microprocessor to apply a first effect to a first frame of said image, said first microprocessor applying said first effect while emulating a second microprocessor;
   - using said second microprocessor to apply a second effect to said first-effected frame, applying said first effect to a next frame by said first microprocessor approximately

during the time that said second microprocessor is applying said second effect to said first-effected frame.

77. The method of claim 76 wherein the first microprocessor is a CPU and the second microprocessor is a GPU.

78. The method of claim 76 where emulation is effected through a virtual machine.

79. The method of claim 76 wherein emulation is effected through translating a GPU program to a CPU program.

80. A computer-readable medium having computer executable instructions for performing the method recited in any one of claims 59, 64 or 76.